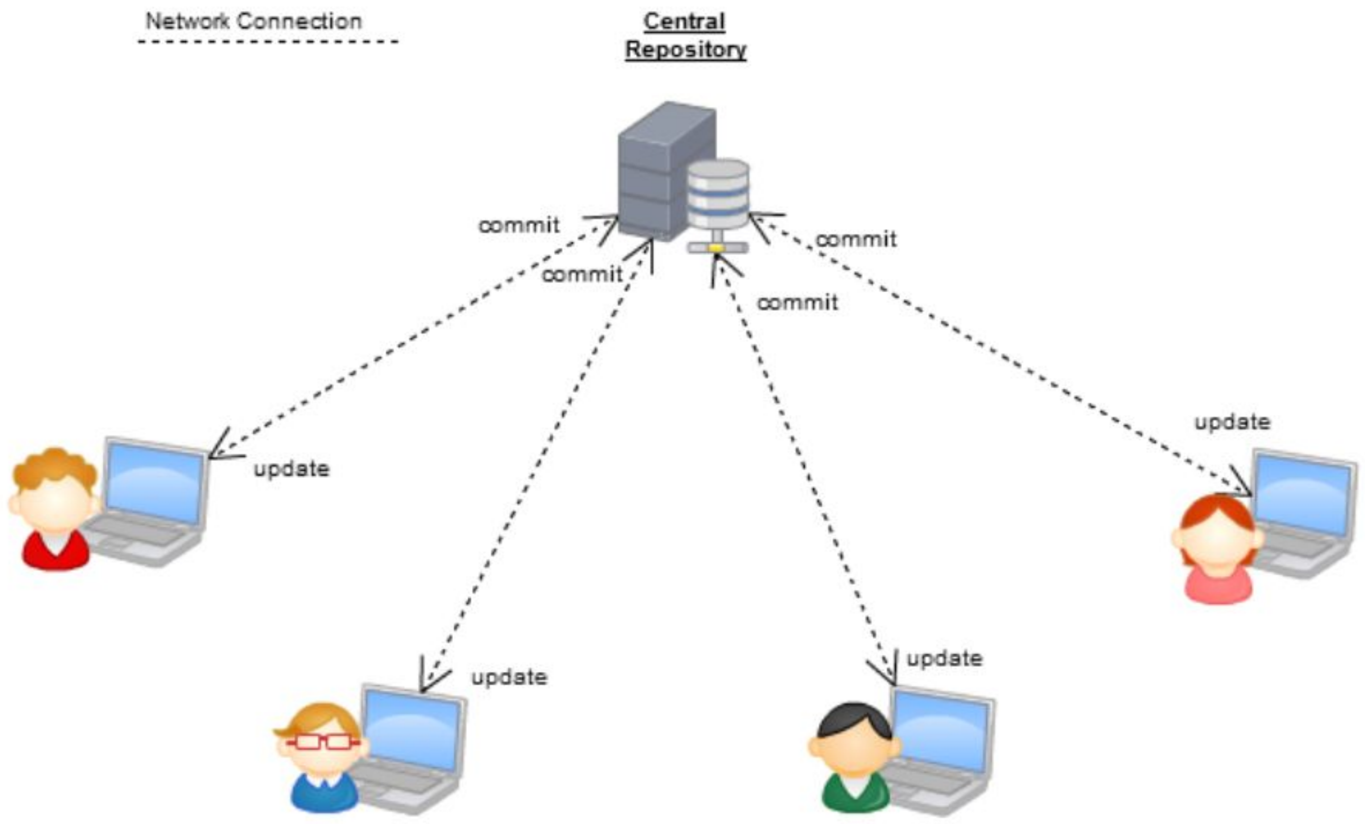
Version Control System



VCS

VCS or Version Control System is a tool that manages different versions of source code. A source code manager (SCM) is another name for version control system. Git is one of the most popular distributed VCS.

CVS

GIT

SVN

Mercurial

Bazaar

Git

Git is a distributed open source version control system commonly used to track and manage changes in the source code. It is a filesystem that lookout for changes in the files including who made the changes.

Git helps us to:

1. Track Document versions.
2. Keep the history of document changes.
3. Encourage teamwork.

GitHub

While Git keeps track of the changes in the code, GitHub is a cloud service that helps teams in communication, collaboration and project management.

GitHub repositories can be private or public. Only the owner and collaborators can view or contribute to a private repositories while public repositories are visible to everyone. With that said, we should be careful of what we publish on your public repositories

Git vs GitHub

git

Version control tool

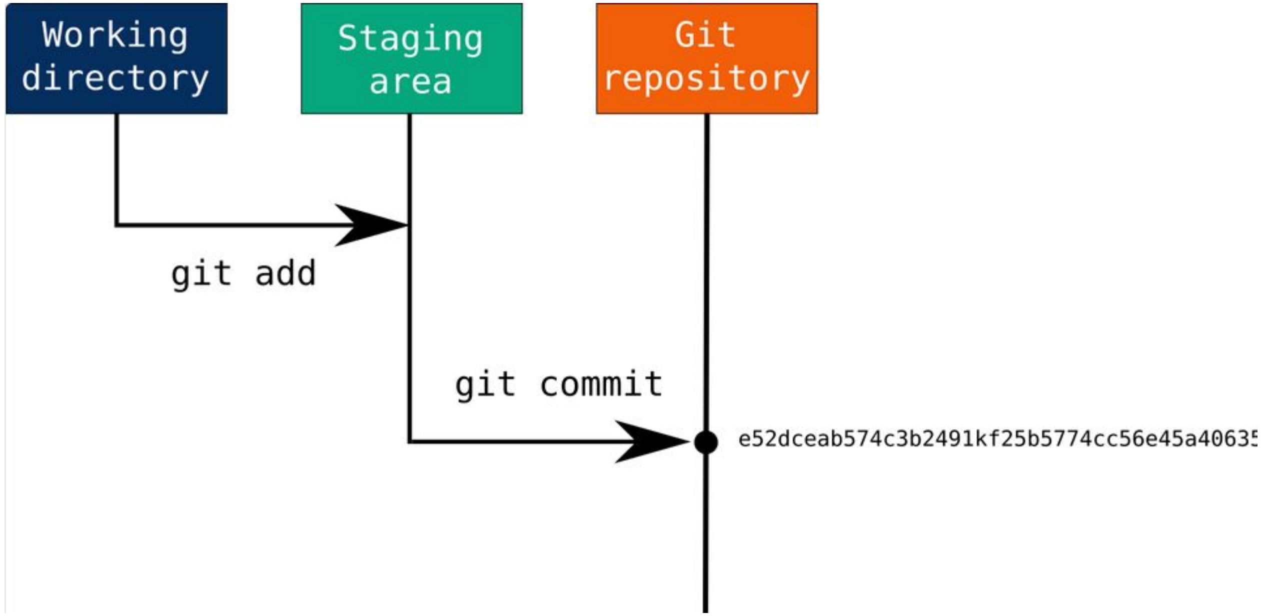
GitHub

Service that hosts Git projects

Terminologies:

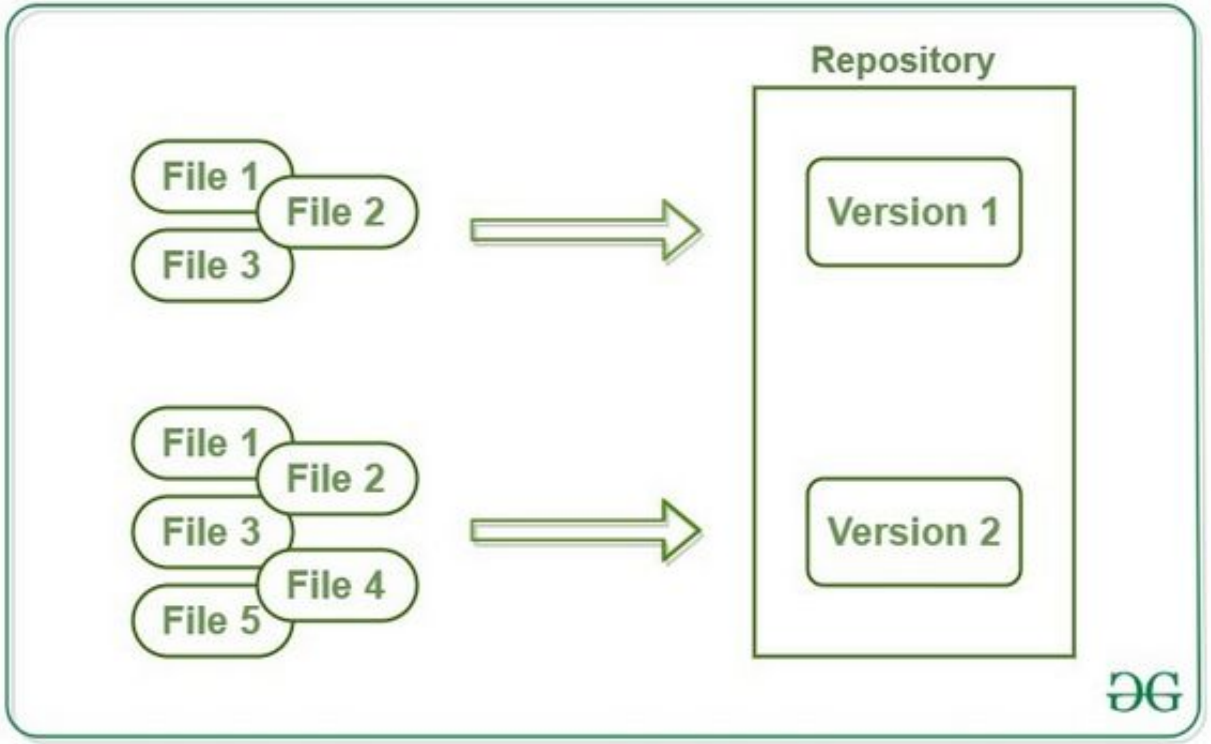
1. Commit – Git Thinks of its data like a set of snapshots of a mini filesystem. Every time we commit or save the state of our project in Git, it basically takes the picture of what our files look like at the moment and stores a reference to that snapshot. We can think of it as a save point in a game.

Everything we do in Git is to help us make commits. So, commit is the fundamental unit of Git.



Terminologies:

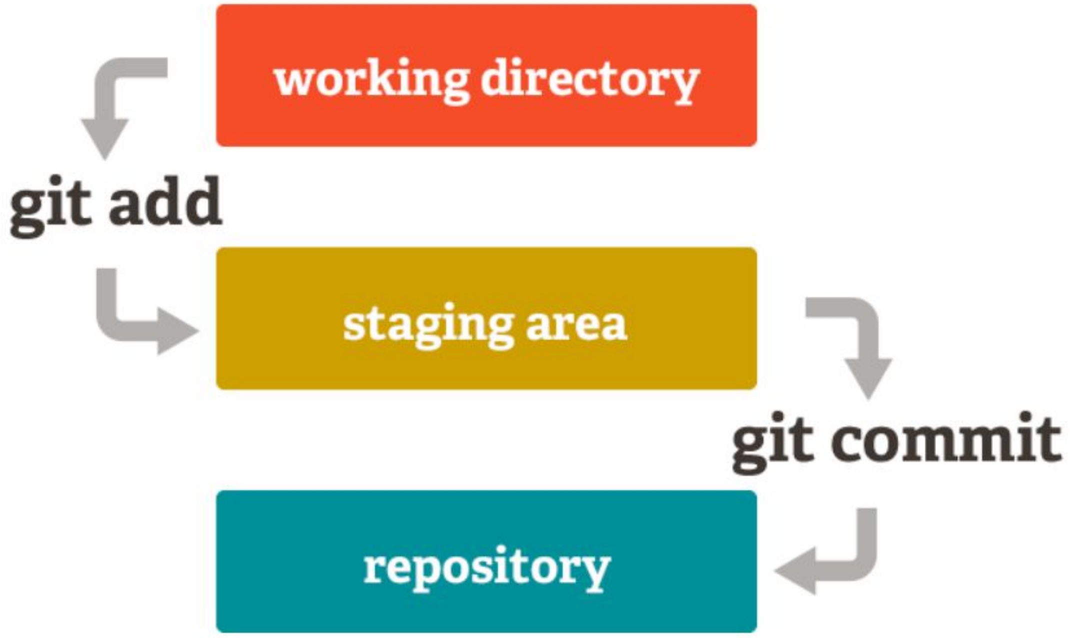
2. Repository/repo – A repository is a directory which contains our project as well as other few files which are used to communicate with Git. Repo can exist either locally on our PC or as a remote copy of another computer. A repo is made up of commits.



Terminologies:

3. Working Directory – Working directory is the files that we see in our computers file system. When we open our project files up on a code editor, we are working with the files in the Working Directory.

This is in contrast to the files that have been committed in the repo.

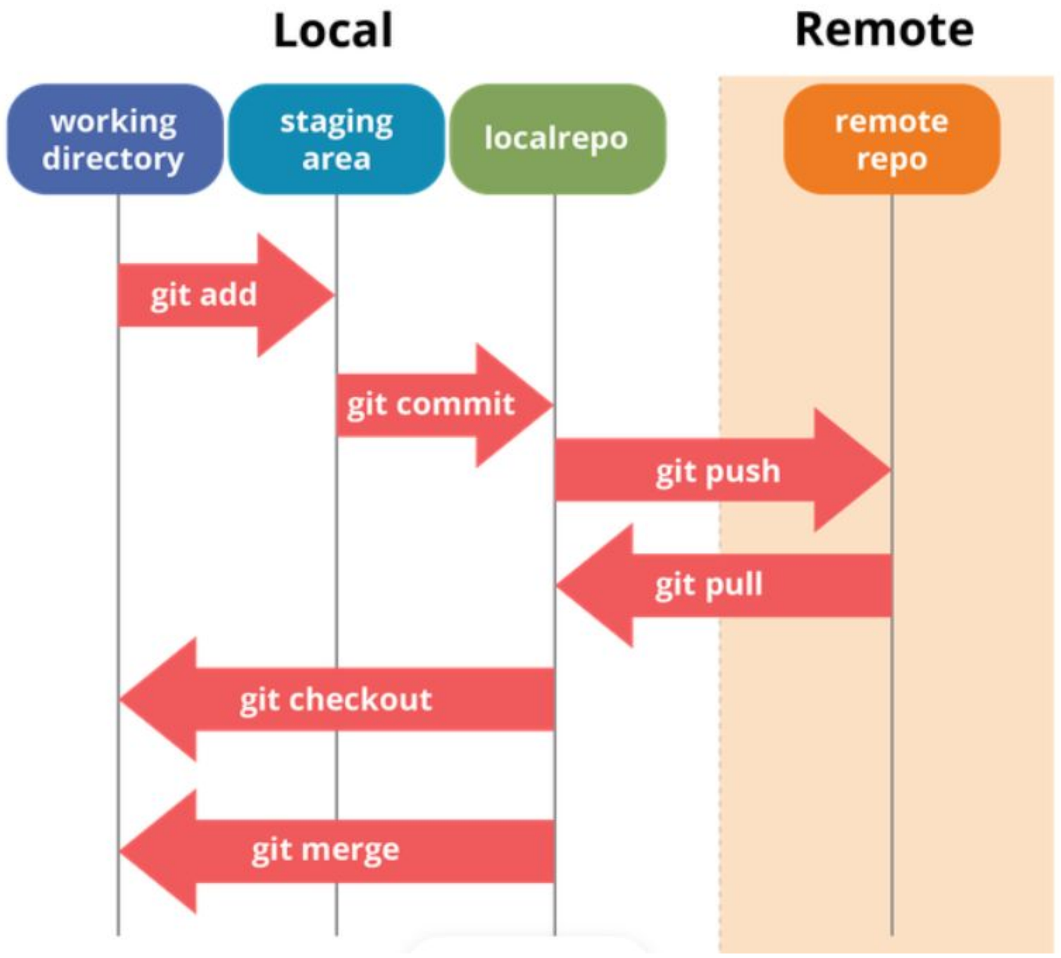


Terminologies:

4. Checkout – A checkout is when content in the repo has been copied to the Working Directory.

Terminologies:

5. Staging Area/Index – A file in the Git directory that stores information about what will go into your next commit. Files in the staging index are poised to be added in the repository.



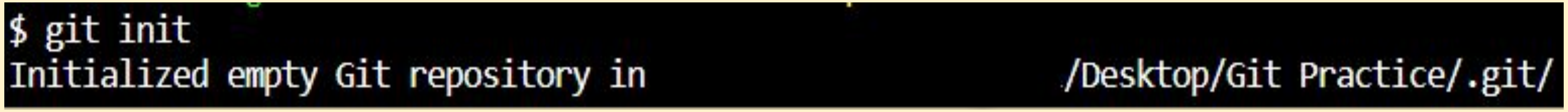
Terminologies:

6.SHA – SHA is basically ID number for each commit.

Here’s what ID for commit might look like :
e2adf8ae3e2e4ed40add75cc44cf9d0a869afeb6

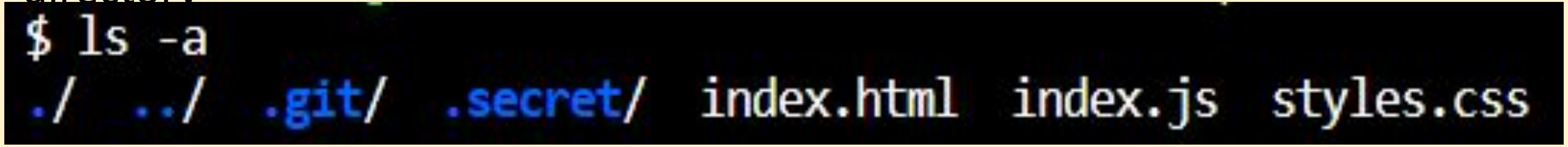
Commands

git init - To initialize git



```
$ git init
Initialized empty Git repository in                    /Desktop/Git Practice/.git/
```

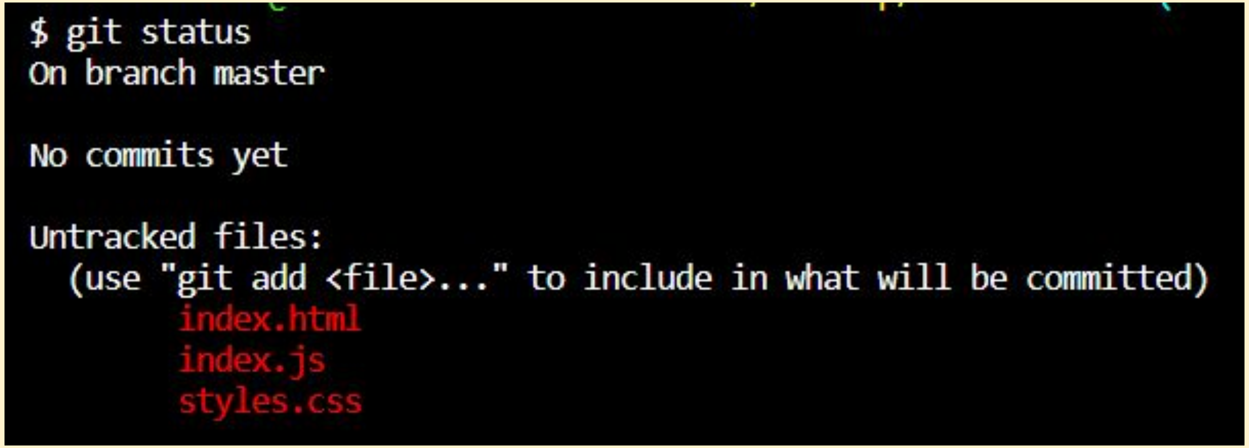
Now, when we enter “ls -a” command, we can see “.git” file in the directory



```
$ ls -a
./  ../  .git/  .secret/  index.html  index.js  styles.css
```

Commands

git status - To see what is currently inside staging area

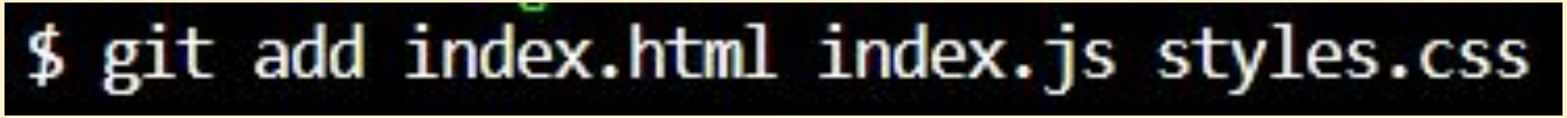


```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        index.js
        styles.css
```

Untracked files will be shown in red and they are the files inside our directory but they are not yet in the staging area.

Commands

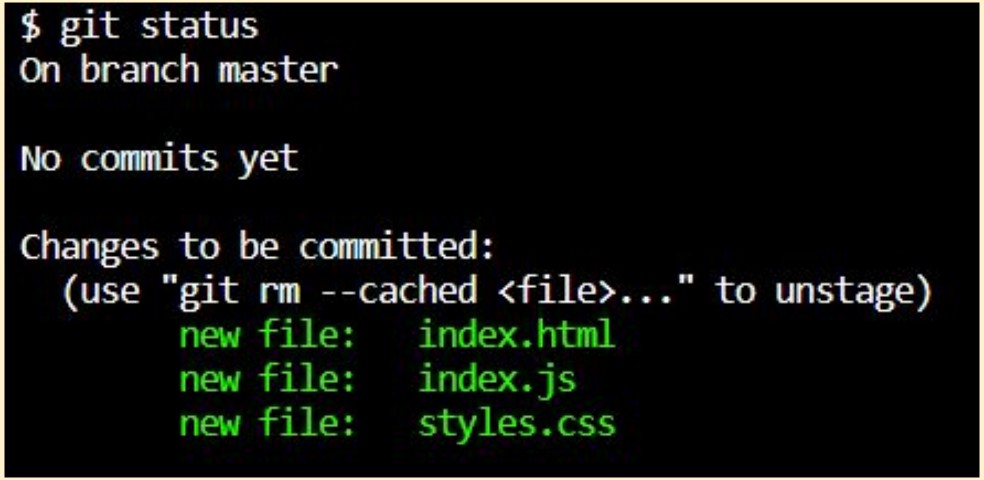
git add <*filename*> - This command is used in order to add the file “<*filename*>” to the staging directory so that we can track changes in the file.



```
$ git add index.html index.js styles.css
```

Commands

Now, again if we check status by using git status

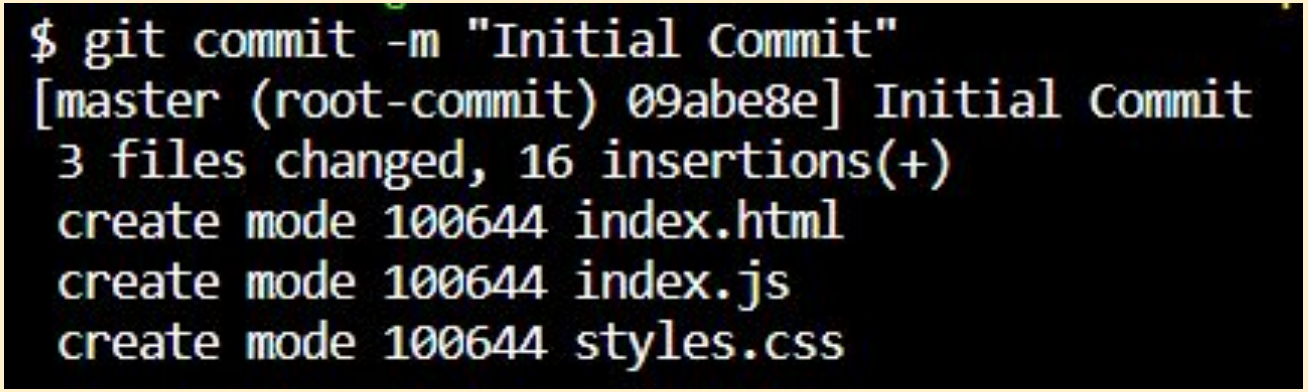


```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   index.js
        new file:   styles.css
```

Here, we can see that the file is added as a new file and the filenames are in green color. This means that the file is now in staging area and is ready to be committed.

Commands

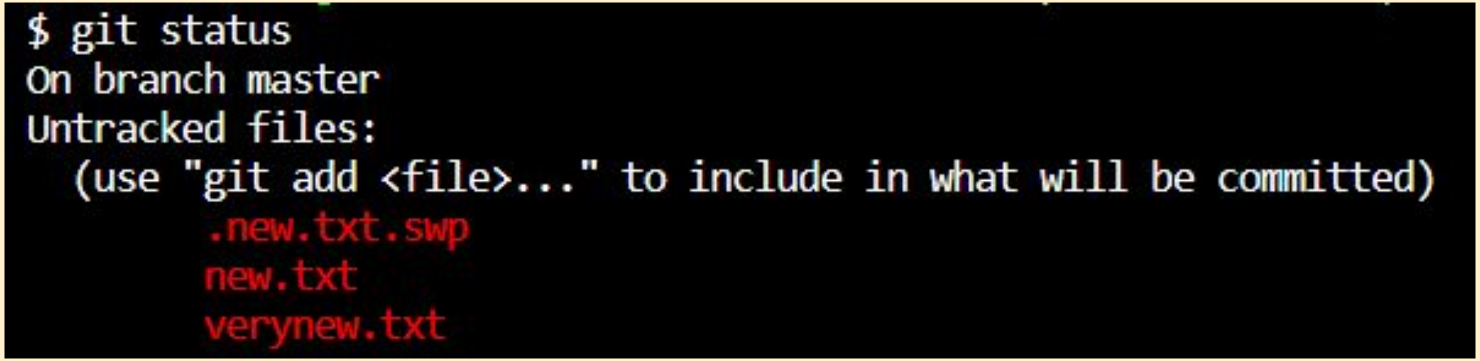
git commit -m “<*message*>” - Commit is used to commit the files under version control. The “-m” flag is used for putting message for that particular commit where the message is written in between the double inverted commas and messages are generally in present tense.



```
$ git commit -m "Initial Commit"
[master (root-commit) 09abe8e] Initial Commit
 3 files changed, 16 insertions(+)
 create mode 100644 index.html
 create mode 100644 index.js
 create mode 100644 styles.css
```

Commands

Now, I created two new text files and added some information on them. The filenames are new.txt and verynew.txt. If we check status again using “git status”, we can see

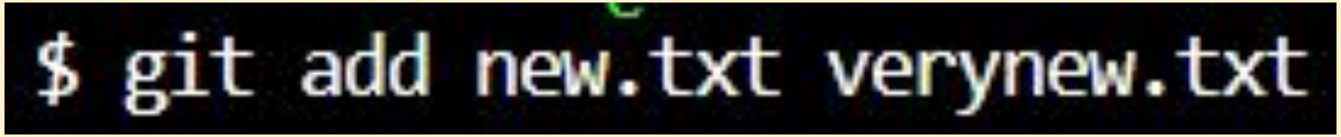


```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .new.txt.swp
        new.txt
        verynew.txt
```

The new files are in working directory but not in staging area, i.e. they are untracked.

Commands

In order to move those files in the staging area, we again use command “git add <*filename*>” in order to track changes and commit files. We can also add all files to the staging area at once by using command “**git add .**”

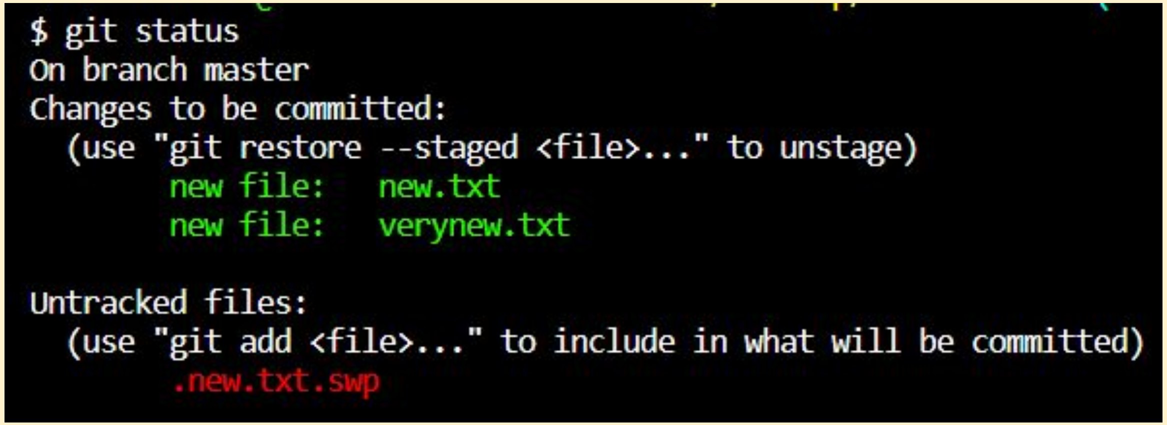


```
$ git add new.txt verynew.txt
```

Now the files are in staging area ready to be committed.

Commands

Now, again if we check status by using git status

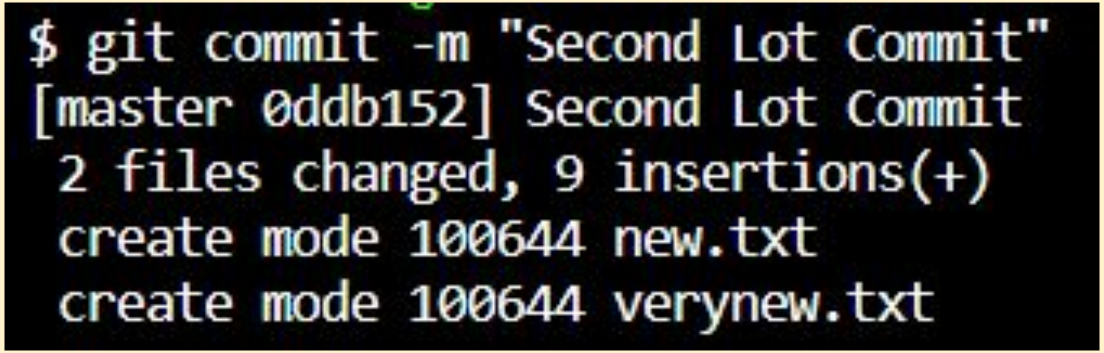


```
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   new.txt
        new file:   verynew.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .new.txt.swp
```

Here, we can see that “new.txt” and “verynew.txt” are ready to be committed.

Commands

We now commit the remaining files.

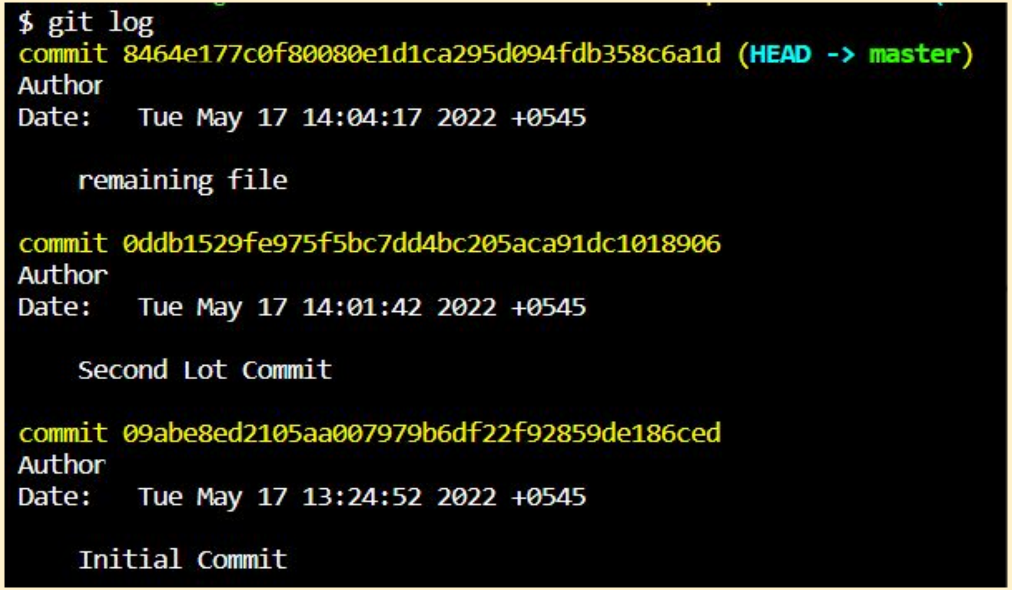


```
$ git commit -m "Second Lot Commit"
[master 0ddb152] Second Lot Commit
 2 files changed, 9 insertions(+)
 create mode 100644 new.txt
 create mode 100644 verynew.txt
```

Here, we can see that “new.txt” and “verynew.txt” are committed.

Commands

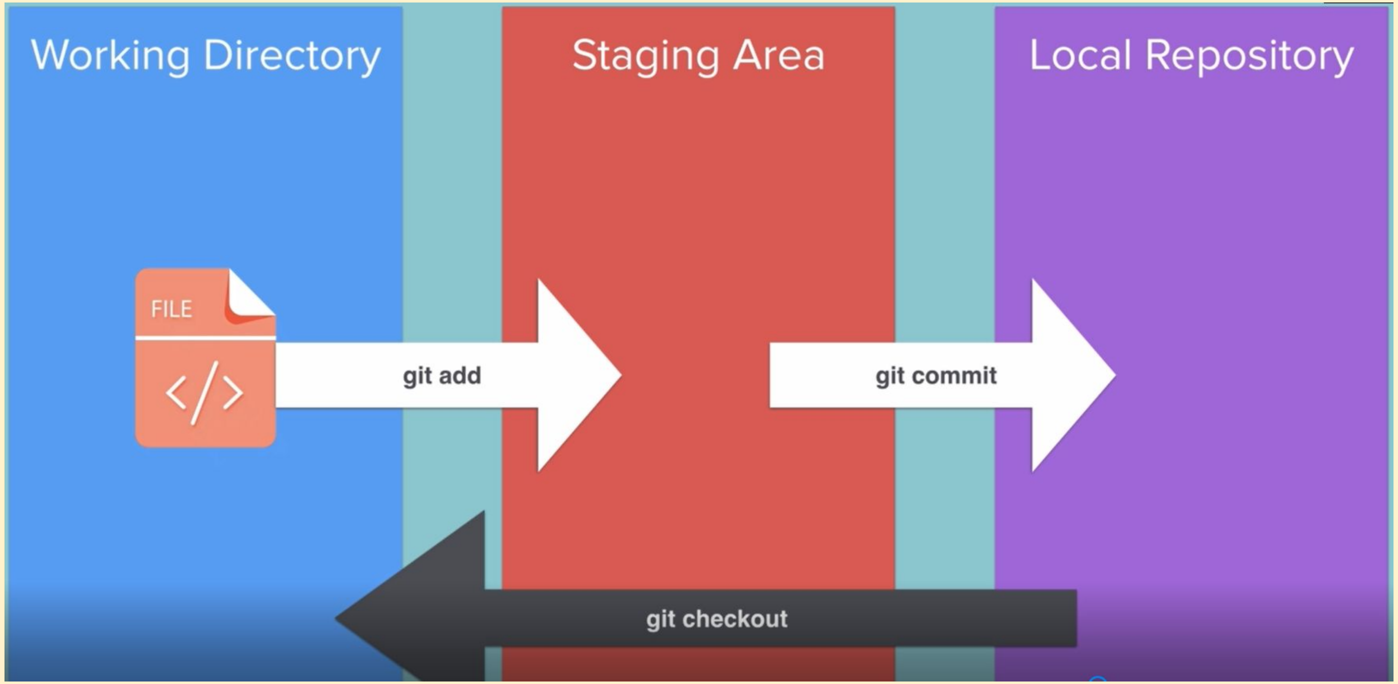
git log - This command is used to view the info about the commits history as shown in the figure below.



```
$ git log
commit 8464e177c0f80080e1d1ca295d094fdb358c6a1d (HEAD -> master)
Author
Date:   Tue May 17 14:04:17 2022 +0545

    remaining file

commit 0ddb1529fe975f5bc7dd4bc205aca91dc1018906
Author
Date:   Tue May 17 14:01:42 2022 +0545

    Second Lot Commit

commit 09abe8ed2105aa007979b6df22f92859de186ced
Author
Date:   Tue May 17 13:24:52 2022 +0545

    Initial Commit
```

(HEAD -> master) represents where we are right now.

Commands


Working Directory
Staging Area
Local Repository
FILE
</>
git add
git commit
git checkout

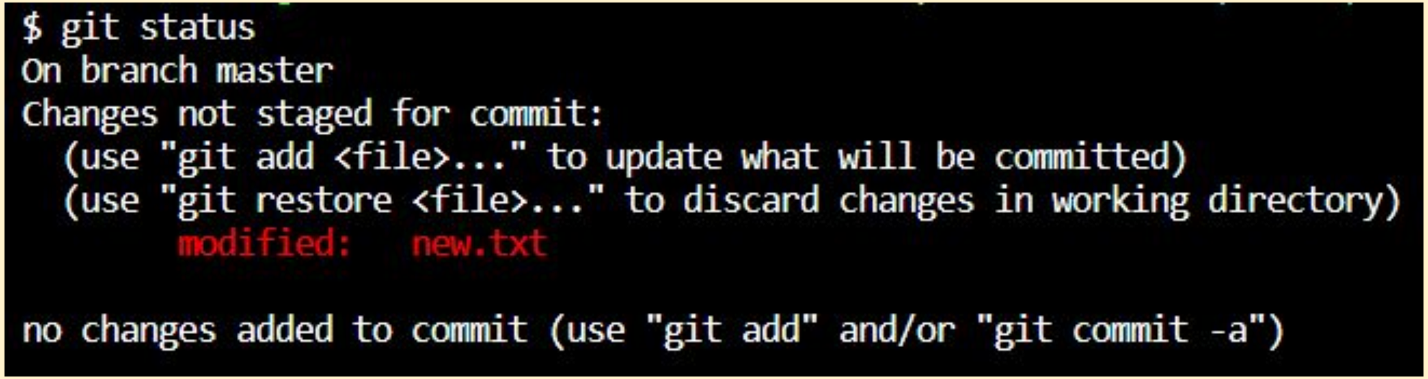
Commands

At the moment, all our files are good and perfect. Now let’s say that we decided to make some changes to our file and completely messed it. How can we get back to the previous position?

This problem can be solved by using command “**git checkout**”.

Commands

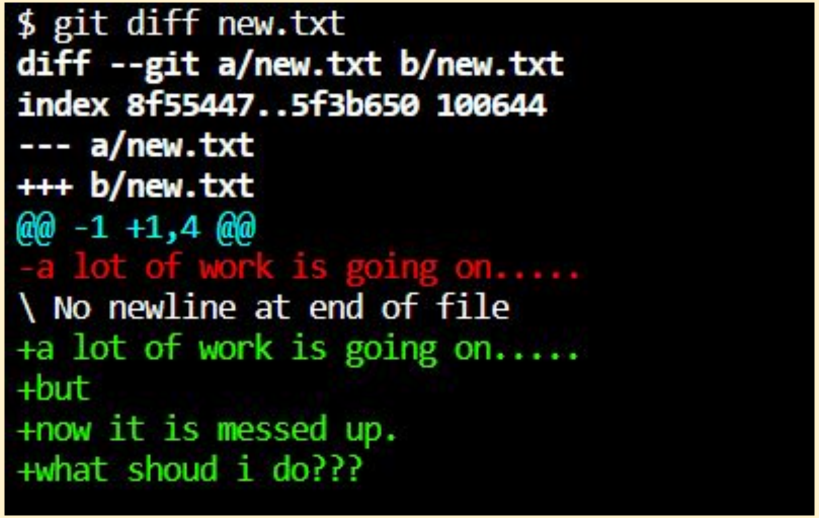
We can use “**git status**” to check that the file is modified locally but we can also see that it is not yet in the staging area.



```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   new.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Commands

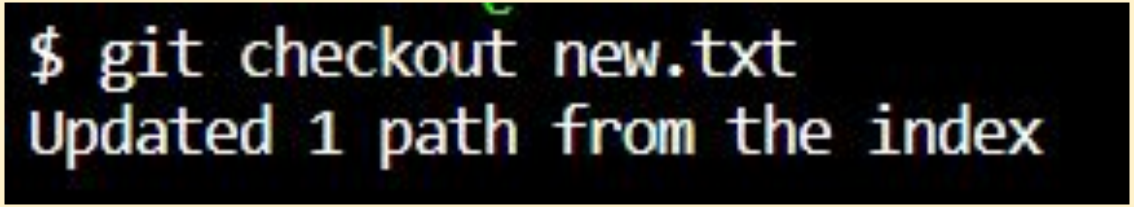
We can also see the differences between the present file and the previous file by using command "**git diff <*filename*>**"



```
$ git diff new.txt
diff --git a/new.txt b/new.txt
index 8f55447..5f3b650 100644
--- a/new.txt
+++ b/new.txt
@@ -1 +1,4 @@
-a lot of work is going on.....
\ No newline at end of file
+a lot of work is going on.....
+but
+now it is messed up.
+what shoud i do???
```

Commands

Now, In order to revert the changes we made and roll back to older version we use command “**git checkout <*filename*>”.**



```
$ git checkout new.txt
Updated 1 path from the index
```

The file is back to it’s older version (last checkpoint).